

Remarks and Arguments

Claims 1-15 and 35-52 have been presented for consideration. Claims 1, 3, 12, 13, 14, 15, 35 and 44 have been amended.

Claim 1 was rejected under 35 U.S.C. §112, second paragraph, for lack of antecedent basis for the phrase “the data maintained by the remote network-capable device” in lines 23-24. In response, claim 1 has been amended, in lines 4-7, to recite that “each of said local network-capable and remote network capable devices” comprises “a memory for storing a local copy of the data...” Claim 1 further recites a data change engine and a dynamics manager and, in lines 23-27, recites “so that the same data changes are made in the same order to the local copies of the data stored in both the local network-capable device and the remote network-capable device.” thus, the rejected phrase “the data maintained by the remote network-capable device” has been deleted and replaced by a phrase that finds antecedent basis in lines 4-7 of amended claim 1. Consequently, the rejection under 35 U.S.C. §112, second paragraph is hereby traversed.

Claims 1-4, 12, 14, 15, 35-38, 40, 41 44-47, 49 and 50 have been rejected under 35 U.S.C. §103(a) as obvious over Jain (previously cited) in view of U.S. Patent No. 5,170,480 (Mohan.) The examiner comments that Jain discloses all of the claimed elements except that Jain does not explicitly disclose the remaking of the undone data changes in another order. However, the examiner claims that the Mohan reference discloses a mechanism for undoing and redoing data changes made to a database and a replica database as part of a forward and a backward recovery process. In this process, the examiner claims, undo records are logged so that undone data changes can be remade in another order. The examiner concludes that it would have been obvious to combine the teachings of Jain and Mohan in order to speed the application of queued records to a second database without locking the resources.

The present invention is concerned with maintaining local copies of data consistent in a distributed peer-to-peer computer system. The term “consistent” means that, when each data change is applied to the local data copy in each computer, that data copy is in the same state that the local data copy of the computer originating the change request was when the request was originated. See the instant specification

page 29, lines 6-27. Consistency requires that the data change requests or “deltas” contain information that allows them to be placed in an absolute order regardless of which computer in the system generated the data change request. In the disclosed embodiment, this information comprises sequence number, endpoint and dependency information. See page 29, line 29 to page 31, line 29.

Jain lacks this absolute ordering. Instead, Jain stores information regarding transactions. Each transaction includes process calls generated by a single computer system. A transaction includes information, such as a delivery order number (DON) and call identifiers (call_ids), that orders process calls, or transactions steps, within that transaction. This is explained in Jain at column 7, line 51 to column 8, line 61. Thus, when a transaction is sent to a remote location, the order information can be used to properly order the transaction steps within the transaction. In addition, Jain indicates that, before a transaction starts, a savepoint can be established so that, if the transaction does not complete, all of the transaction steps can be rolled back.

However, the transaction step ordering information is not particularly useful if a transaction generated by one computer conflicts with a transaction generated by another computer. In this case, Jain is limited to raising an exception if such a conflict occurs, rolling back the transaction in progress or passing control to an application so that multiple conflict routines (which are unspecified) can be run. See Jain column 21, line 61 to column 22, line 60. The lack of absolute ordering has a profound difference in the operation of Jain as opposed to the present invention.

For example, consider a situation where there are two computers C1 and C2, and each has a value for a quantity on hand (qoh – the terminology used in Jain) of a widget. Assume that, at the beginning of the example, both computers are in the consistent state where the value of qoh is 80. Now, assume that simultaneous transactions, T1 and T2, occur on C1 and C2, respectively. T1 is an order for 50 widgets, so it changes the qoh value to 30 on C1. T2 is an order for 60 widgets, so it changes the qoh value to 20 widgets on C2.

As mentioned above, in accordance with the principles of the invention, there is an absolute ordering of the transactions. Assuming T1 is ordered before T2, the following would happen. On C1, T2 would be received and executed. It would be up to

the logic in the data-change engine in C1 to deal with the fact that the transaction T2 was attempting to subtract 60 widgets from a qoh of 30. An exemplary implementation of this logic would be to check the qoh before subtracting the transaction quantity and, if the qoh is not large enough, then mark the order in T2 as unfilled and leave the qoh at 30.

On C2, in accordance with the invention, when T1 arrives, the dynamics manager will detect that T1 is ordered before T2 and will realize that T2 must be rolled back because it has already been executed. This rollback will restore the qoh to 80. Then T1 will be executed, and the qoh will be set to 30. Now T2 will be rolled forward. The engine logic will be executed with the data in the exact same state that it was when T2 was executed on C1. In the exemplary implementation, the data-change engine logic will leave the qoh at 30 and mark the order unfilled. Thus, the state of the qoh in C2 is in the identical state of qoh in C1. What is important is that at the time the data-change engine executed each transaction on each computer, the data was in the identical state.

Jain discloses both row-level and procedure-level replication. First considering row-level replication, in the above example, T1 and T2 would each consist of an update operation. Both operations have the old (80) and new values (30 and 20, respectively) as parameters. On C1, when T2 arrives it will be executed. The Jain logic will check the current value against the old value parameter (as disclosed in Jain Figure 5B1 step 536) and realize that the values are not the same (current value 30, old value 80). Assuming lost updates should be prevented (as set forth in step 536 in Figure 5B2) then an exception will be logged (step 540) and the qoh value will remain 30. On C2, transaction T1 will arrive. Again the current value will not match the old value, so an exception will be logged and the value will remain 20. Thus, the two computers are not in a consistent state and there are exceptions that need to be resolved. Jain discloses no mechanism for resolving these exceptions.

Procedure level replication in Jain results in a similar outcome. In the procedure level replication process as set forth in Jain it would be the responsibility of the implementer of the procedure to ensure data consistency. For example, consider a simple procedure that works just like the data-change engine logic as described above.

In accordance with this procedure, when T2 arrives at C1, the procedure would realize that there are not enough widgets on hand to fulfill the order in T2. Consequently, it would leave qoh at 30 and mark the order in T2 as unfilled. The procedure would probably also want to log an exception. When T1 arrives at C2, the procedure will also realize that there are not enough widgets, leave the qoh at 20 and mark the order in T1 as unfilled. Again, the procedure would probably want to log an exception. Like in the row level replication case, C1 and C2 are now in inconsistent states and there are exceptions to handle.

The Mohan reference describes a database replication scheme where changes are made to the database only on one computer. The change sets are then replicated through the use of multiple change queues. When placing the changes into the queues, the changes are broken into sets of non-conflicting operations by insuring that REDO records associated with a single database page are placed on the same queue (see Mohan column 4, lines 57-64.) Since the operations are non-conflicting, they can be applied to the replica in any order, or simultaneously. Thus, the changes can be applied in parallel to different data thereby decreasing the time required to make the changes. This is significantly different than changing the order of applied transactions that affect the same data in order to achieve a consistent ordering as the present invention does during rollback and rollforward of the data changes. In fact, Mohan teaches away from the present invention because it teaches that the data changes should be arranged so that they can be applied in any order. The present invention teaches that the data changes must be applied in an absolute order, undoing and redoing the changes, if necessary, to achieve that order.

The combination of Jain and Mohan cannot teach or suggest the invention because neither reference teaches that data changes must be applied in an absolute order regardless of whether the data changes are generated on the local computer or a remote computer and that the changes (no matter where they are generated) must be undone and redone to achieve that order. In particular, as mentioned above, Jain does not teach a mechanism for handling conflicts between a transaction generated on one computer and a transaction generated on another computer. Mohan cannot change this teaching because it teaches that the changes should be arranged so that conflicts do not occur. Thus, the combination of Jain and Mohan might at best teach that the

transaction steps of Jain could be applied in parallel without a conflict occurring between those steps.

Claim 1 has been amended to particularly point out these differences. For example, in lines 9-10, amended claim 1 recites that the data change requests have an absolute order relative to each other. Amended claim 1 further recites a dynamics manager that causes the making of received data changes (from the local and remote devices, lines 12-13) in the absolute order (lines 17-18), and, responsive to a data change request being received out of the absolute order, the undoing of the selected data changes to a point where a data change corresponding to the out-of-order data change request should have been made (lines 19-22), the making the out-of-order change (line 22) and the remaking of the undone data changes in the absolute order so that the same data changes are made in the same order to the local copies of the data stored in both the local network-capable device and the remote network-capable device (lines 22-27.) As set forth above, the combination of Jain and Mohan does not teach or suggest undoing and redoing data changes received from different computers in order to achieve an absolute order. Consequently, amended claim 1 patentably distinguishes over the cited combination of references.

Claims 2-4 are dependent, either directly or indirectly, on amended claim 1 and incorporate the limitations therefore. Therefore, they also patentably distinguish over the cited references. In addition, claim 3 has been amended to recite that the request sequence number defines an absolute order of data change requests relative to each other. as discussed above, the combination of the cited references does not teach or suggest this. Consequently, amended claim 3 patentably distinguishes over the cited references for that reason.

Claims 12, 14, 15, 35 and 44 have been amended in a similar fashion to amended claim 1 and, consequently, distinguish over the cited references in the same manner as amended claim 1.

Claims 36-38, 40 and 41 are dependent, either directly or indirectly, on amended claim 35 and incorporate the limitations therefore. Therefore, they also patentably distinguish over the cited references. In addition, claim 36 recites placing data change requests on a holding queue when they require that another change was previously

made and that other change was not made. This is a dependency situation where some data change requests are dependent on other data change requests having been made. Neither Jain nor Mohan address this situation. The examiner points to Jain column 8, lines 58-61 and column 20, lines 1-6 as disclosing a call_id or a DON identifier that can be used to determine the order of transactions steps. While both the Jain call_id and DON numbers can be used to order transaction steps, Jain does not disclose placing the changes on a holding queue until a previous step is completed as recited in claim 36. Mohan also does not disclose such a holding queue. This is because both Jain and Mohan refer to transactions generated on a single computer. Therefore, all of the steps in such a transaction are present and the situation recited in claim 36 does not arise. The lack of a mechanism for introducing a dependency in the data change requests causes problems with the Jain system.

Claims 45-47, 49 and 50 are dependent, either directly or indirectly, on amended claim 44 and incorporate the limitations therefore. Therefore, they also patentably distinguish over the cited references. In addition, claim 45 recites placing data change requests on a holding queue when they require that another change was previously made and that other change was not made in the same manner as claim 36.

Claims 5-11, 13, 39, 42, 43, 48, 51 and 52 have been rejected under 35 U.S.C. §103(a) as obvious over Jain in view of Mohan and further in view of Niblett (previously cited.) The examiner asserts that Jain and Mohan disclose the recited limitation except for the "endpoint" recited in the listed claims. The examiner claims that Niblett discloses such endpoint numbers and their use as identifiers. The examiner concludes that it would have been obvious to combine Jain, Mohan and Niblett in order to more quickly apply serialize updates.

As mentioned in the response to a previous office communication, the Niblett patent discloses a token passing conference ring network in which a "permit-token" is used to serialize updates. When updates are performed, the permit-token is used to establish an order for the updates. In this manner, each node in the system will know when an update is missing. Consequently, the node can wait until the missing update has been received before applying later updates. This operation is clearly set forth in the Niblett abstract.

Claims 5-11 depend on amended claim 1 and incorporate the limitations thereof. The combination of Jain, Mohan and Niblett references does not suggest the make, undo and remake operations that are recited in amended claim 1 to establish the required absolute order. As discussed above, Jain and Mohan do not teach or suggest the recited combination. The addition of Niblett does not change this conclusion. This is because the Niblett serialization mechanism guarantees that an update cannot arrive "out-of-order" after other updates have been applied as in the present invention. Therefore, there is no reason to undo updates and then redo them in the absolute order as recited in amended claim 1 as discussed above and Niblett does not mention any such operation. Consequently, the combination of Jain, Mohan and Niblett cannot teach or suggest the recited limitations since no of the reference teaches or suggests the recited combination. Therefore, claims 5-11 patentably distinguishes over the cited combination of Jain, Mohan and Niblett.

Claim 13 is dependent on amended claim 12 and incorporates the limitations thereof. Since amended claim 12 recites essentially the same limitations as amended claim 1, claim 13 distinguishes over the cited combination in the same manner as amended claim 1.

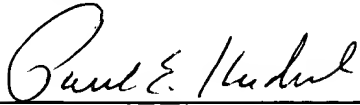
Claims 39, 42 and 43 are dependent on amended claim 35 and incorporate the limitations thereof. Since amended claim 35 recites essentially the same limitations as amended claim 1, claim 35 distinguishes over the cited combination in the same manner as amended claim 1 and therefore claims 39, 42 and 43 also distinguish over the cited references.

Claims 48, 51 and 52 are dependent on amended claim 44 and incorporate the limitations thereof. Since amended claim 44 recites essentially the same limitations as amended claim 1, claim 44 distinguishes over the cited combination in the same manner as amended claim 1 and therefore claims 48, 51 and 52 also distinguish over the cited references.

In light of the forgoing amendments and remarks, this application is now believed in condition for allowance and a notice of allowance is earnestly solicited. If the examiner has any further questions regarding this amendment, he is invited to call applicants' attorney at the number listed below. The examiner is hereby authorized to

charge any fees or direct any payment under 37 C.F.R. §§1.17, 1.16 to Deposit Account number 02-3038.

Respectfully submitted



Date: 7/26/04

Paul E. Kudirka, Esq. Reg. No. 26,931
KUDIRKA & JOBSE, LLP
Customer Number 021127
Tel: (617) 367-4600, Fax: (617) 367-4656